WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Cyber Safety Analysis Using Reverse Engineering

Syed Arif Islam[a], Dr. M. Mohan Kumar[b]

[a]*Department of Computer Science, Karpagam Academy of Higher Education (KAHE), Coimbatore, 641021, India.*
[b]*Department of Computer Science, Karpagam Academy of Higher Education (KAHE), Coimbatore, 641021, India.*

A B S T R A C T

Cybercriminals frequently employ malware, commonly referred to as harmful software, to further their objectives by monitoring online activity, seizing private data, or restricting access to computers. Attacks by malicious software and risks to information security are now complicated processes. Due to the variety and volume of these attacks and threats, a range of defence mechanisms have been developed. Unfortunately, current detection technologies are unable to keep up with the new strategies used by malware developers to evade anti-malware programmes. One of the finest strategies to stop and a potent weapon in the ongoing battle against cyberattacks is reverse engineering. Reverse engineering can be used to complete a number of cybersecurity-related tasks. In this paper, we present a combination of techniques to speed up and improve malware detection processes; enable malware detection systems to find malware with high precision and in less time; and assist network security experts in responding effectively because early detection of security threats is crucial in defending against attacks.

Keywords:Reverse Engineering, Attacks, Malware, Cyber

1. INTRODUCTION

Disassembling a thing is the act of reverse engineering. It is done mostly to learn how something works and to analyse it, although it is frequently used to reproduce or improve the object. Reverse engineering can be used to analyse software, hardware, military technologies, and even biological processes. Reverse engineering is a technique used in earlier industries and applied to computer hardware and software. The machine code, or string of 0s and 1s that a programme sends to the logic processor, is the focus of software reverse engineering. The machine code is converted back into the original source code using programmer language statements. Reverse engineering can be used to teach someone how something works, repurpose outmoded goods, do a security study, obtain a competitive advantage, or any number of other things, depending on the technology. Reverse engineering is the process of extracting information from a final product regardless of how the knowledge is applied or what it relates to. Reverse engineering is a technique used to determine how a system or thing operates. There are numerous justifications for doing this. Reverse engineering can be done to figure out how something operates, then reproduce it or make a similar object with improved functionality. Reverse engineering software or hardware frequently aims to find a cheaper way to make a similar product or is done since the original product is no longer being produced. In the field of information technology, reverse engineering is also used to solve compatibility problems and make the hardware or software operate with other hardware, software, or operating systems that it wasn't designed to.

Cyber defencecentres are very important in many nations due to the ongoing expansion of malware, which has led to the creation of massive dangers in information and security points. Virtual space is vulnerable to threats from several sources, including smugglers and contraband, just like physical borders [1]. The majority of these attacks, according to experience, are caused by malware. Resource protection greatly depends on the timely detection of virtual space security assaults. We should use techniques for identifying good and bad software behaviours to be able to identify which software is troublesome and which ones are not in order to detect such malwares before the appearance of dangerous consequences. To avoid issues with the detecting procedure, we should thus evaluate both types of software [2]. Fig.1 shows an increase in malware volume between 2016 and 2022. It could see an increase in assault activity in the next years.

* *Corresponding author.*
E-mail address: syedarifislam@gmail.com

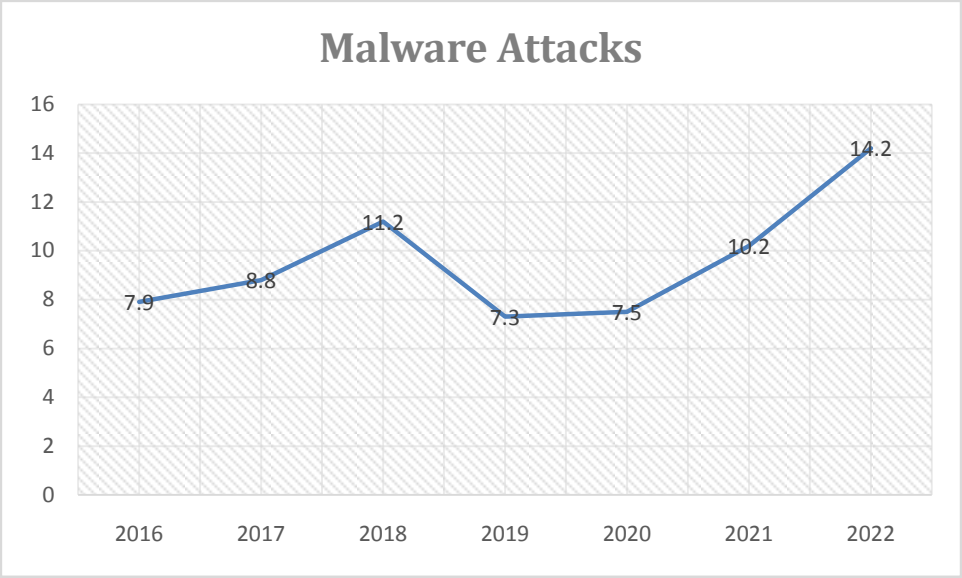

Fig. 1Number of malware attacks.

Malware researchers need a broad range of skills, which are typically acquired over time through practise and self-education. Reverse engineering (RE) is one of the most sophisticated skills a researcher may possess and is a crucial component of malware investigation and study. One of the causes of the absence of RE personnel in businesses is this. Many novice researchers find it challenging to begin their careers in RE. A researcher may stand out if they have a strong foundation in RE with the necessary expertise and terminologies. Fundamentals of RE for malware researchers, analysts, and IR professionals with no prior experience Understanding a compiled program's capabilities and functionalities is known as RE in software. The goals of RE vary, ranging from researching compilers and vulnerabilities to recovering lost source code and enhancing performance. In order to address threats and research various malware families, malware RE focuses primarily on comprehending the capabilities and operations of malware. RE can take a lot of time. Usually, don't start reversing a trojan right away while researching it. Instead, triage malware analysis should be carried out by running the malware in a sandbox, extracting strings, and other methods. If more context is required, it might be provided by this early malware analysis phase of RE. For example, one can anticipate seeing a specific capability that the malware demonstrates or search for specific strings in the disassembler. The malware's Command and Control (C2) could fail, the malware's configuration could depend on a file that is not present on the machine, the malware has the ability to evade sandboxes, or the malware would only operate in a specific environment. To do this, RE, a component of sophisticated static malware analysis, is far more successful.

2. OVERVIEW OF MALWARE

All Malware and victim systems are terms used to describe computer programmes that are damaging in nature and apply to intruding systems [3]. The term "malware" refers to any virus, worm, trojan horse, and other programme designed to distract users and violate privacy. Malware is typically divided into different types based on how it behaves and how it attacks. For instance, the following list classifies some types of malware: Each type of malware—a virus, worm, spyware, and rootkit—has a distinct behaviour that is detailed below:

2.1. Trojan Horse

The term "Trojan" refers to a programme that pretends to be a legal piece of software but, when downloaded and used, embeds harmful code or files on the host [4]. Most often, when a trojan horse is performed, it will install a virus or possibly nothing at all. It is not capable of self-replication and needs the system administrators to turn it on. However, it has the potential to grant remote access to an attacker, who may then engage in any harmful behaviour that would be of interest to them. Depending on the payload are tied to, trojan horse programmes can impact the host in a variety of ways and are typically disseminated through social engineering [5].

2.2. Viruses

A virus is a group of harmful programmes that joins a host application. The malicious code runs when the host programme is executed, which is a requirement for the host application to function. By looking for additional host programmes to infect with the malicious code, the virus strives to replicate itself. The virus eventually turns on and releases its payload. A virus's payload typically causes harm. It could also enable backdoors that attackers can

employ to remotely access systems, remove files, induce arbitrary reboots, connect the machine to a botnet, or join one. Some older viruses might occasionally just display a message, such as "Legalize Marijuana!" Most viruses don't start doing harm right away. They allow the virus to begin reproduce by doing this. The virus is frequently (though inadvertently) executed by a user, but it can also be started automatically by an operating system when a user interacts with it. For instance, the system can run the virus that has infected it when a user plugs in a USB device that is infected. Keep in mind that not all malware requires human involvement to function. Worms are one such self-replicating organism that does not require user interaction.

2.3. Rootkit

Malware called rootkit can conceal both its own presence and its activity on a target machine. Without the system owner's knowledge, the rootkit owner is able to run files and settings on the victim machine. Typically, it runs alongside the original operating system core files that it attaches to. In order to alter performance trends and the outcomes of their operation, rootkits attempt to attack the operating system's original structures, programmes, and the integrity of the contents. The following techniques can be used by rootkits to conceal themselves from users: A rootkit may access all system requests, including reading files and launching programmes, by integrating its codes with low-level operating system code. Rootkit injects its malicious code into legitimate processes, allowing it to access the memory and run its harmful programmes [6].

2.4. Vampire Virus

Researchers generally try to reverse engineer the code when they find a new infection. Programmers that create programmes, first create them in a computer language like C, C++, or C#. Despite having specific syntactic requirements, these are simple for speakers of the language to read. The code is subsequently translated into an executable programme by developers. Decompiling the executable application and examining the code to see what it performs is known as reverse engineering. Armored viruses employ a number of strategies to make it more challenging for AV researchers to reverse engineer them. Armoured viruses employ several techniques, including:

- Cryptic Code: Some viruses with armour employ cryptic coding that is intended to hide true intentions.
- Encryption: Some compilers encrypt the virus's code, which makes it more challenging to decompile. Before it can be decompiled, this code must first be encrypted.
- Hiding: Some viruses try to conceal true location by making antivirus software believe the file is somewhere else.

2.5. Logic Bomb

A logic bomb is installed on a system and waits for an external event, example data input or reaching a specific date, to create, delete, or even edit a special file that will cause the system to be damaged [7].

2.6. Backdoors

A backdoor is a type of software that accesses a computer system without authorization and accomplishes its objectives without going via the standard system entry points [8].

2.7. Adware

The term "adware," which stands for "advertising supported software," refers to software that automatically displays adverts, particularly pop-up ads on websites. Most of them are created so that advertising may use them to generate income. Adware that includes spyware can be highly harmful since it can track user activities and steal personal data [9].

2.8. Spyware

This is a malicious programme that makes use of operating system features with the goal of monitoring user behaviour. They occasionally possess extra powers, such as the ability to change the security settings on the infected system or interfere with network connections. By affixing themselves to trustworthy software, trojan horses, or even by taking advantage of well-known software flaws, they spread. Spyware can track user behaviour, gather keyboard data, and internet usage patterns before sending the data to the program's creator [10].

Understanding a suspicious file's or URL's behaviour and intent is the goal of malware analysis. This analysis describes the procedure used to examine and ascertain the function and goal of a certain malware sample. By removing information from malware analysis, we are able to gather the data we need to create efficient detection methods for harmful code. It offers a fix for high loyalty warnings earlier in the attack life cycle and gives knowledge of the different malware categories and attack tactics.

Because it frequently requires getting through barriers that baffle researchers or obfuscate the purpose and origin of the infection, manual malware analysis can be time-consuming. Reconstructing a series of steps required to analyse a certain malware's operation is a common task in manual malware analysis. Another technique to examine whether malware is present on a system is reverse engineering. This study cannot be carried out in production

systems that are compromised by malware, such as an enterprise or test system. Unquestionably, it is a useful technique for identifying malware, such as a malicious executable file, that may be present on a machine.

3. LITERATURE SURVEY

Reverse engineering encompasses a wide range of tasks, such as system data analysis and the decompilation and disassembly of executable files and libraries. Reverse engineering is a technique used in computer security to analyse malware activities and develop solutions to stop it. On some extremities of the spectrum, reversing is frequently utilised in relation to harmful software. Both antidote creators and criminal programmers use it. Reversing is therefore common among crackers, who use it to analyse and then get rid of various copy of protection measures. Reverse engineering malware is a common practise among companies that create security software. Instead of reactively creating defences against particular dangerous programmes, a cybersecurityorganisation might establish techniques to mitigate the tactics used by hackers by recognising and studying a piece of malware. For the purpose of finding software security flaws, reverse engineering is frequently performed. Although some companies utilise this to set up defences against these security flaws, malevolent software developers may use this technique to find security flaws that can exploit. Every link in the malicious software chain uses reverse engineering to some extent. Reverse engineering is a common technique used by malware developers to identify faults in operating systems and other programmes. These flaws could be leveraged to breach the system's security measures and enable exploitation, usually through the web. In addition to infecting a system, criminals frequently utilise reversing mechanisms to identify software defects that let malware programmes access private data or even take over a system. Anti-malware programmers check and evaluate any dangerous code that comes into their possession at the next link in the chain. They follow every step the programme makes using reverse engineering techniques to ascertain the potential harm, the projected sickness rate, the best way to remove it from compromised systems, and whether exposure can be fully avoided. The majority of businesses and people have already migrated their data from local storage to cloud-based storage, which provides various operational and protection benefits but is not completely secure. Even the most secure cloud storage services, like iCloud, can't completely secure information, and attackers skilled in reverse engineering can take advantage of the most secure safeguards for cloud-based services. As users upload more content to cloud storage, additional cloud interfaces are created to improve the user experience, which exacerbates the problem. These enhancements create a new potential vulnerability and increase the risk of user information being misused. These are the main issues with reverse engineering in modern cyber security.

4. METHODOLOGY

The research design strategy used in this study adheres to kitchenham and charters' principles and utilises qualitative research [11]. Paper is gathered and processed using systematic mapping study for publication searching. In essence, snowballing is used to choose papers. We conducted my search using a few keywords. Search terms for these publications include "reverse engineering," "cybersecurity," "virus," and "risk." the use of and/or/not. We created a search string utilising those terms that was used in many databases, including:

- IEEE .
- ACM Digital Library.
- SpringerLink.
- Elsevier.
- Google Scholar.
- Wiley and
- Scopus.

Find journal articles that were published in the english language between 2018 and 2022 by doing the search again. Search engines turn out 1,137 useful articles that are connected. 25 publications were then discovered to have appeared in reputable journals or conferences. We submitted a request, and exclusion criteria suggested in this study by Kitchenham et al. Following that, we have incorporated 21 research as significant findings from the literature into this study. In order to analyse the data gathered, we also employed an open coding technique. The findings from the literature were examined by both researchers, which lessened the validity risks.

5. RESULT

A software programme must be disassembled (and occasionally decompiled) in order to be RE as malware. Engineers can examine what a programme does and what systems it affects by converting binary instructions to code mnemonics (or higher-level constructs) using this approach. Engineers can only develop remedies that can lessen the program's intended malicious consequences by being aware of all of its specifics. RE will employ a variety of tools to ascertain the purpose and method of operation of a programme. The reverser would be able to determine the vulnerabilities the software intended to exploit as a result of doing this. Although malware writers are known to leave behind bogus trails, RE can extract signals that disclose when a programme was produced, what embedded resources it may be accessing, encryption keys, and other file, header, and metadata elements. Engineers frequently employ a variety of tools to reverse malware code. Here is a list of a few of the more significant ones:

- Extraction of information: In order to reverse-engineer an object, information regarding its design must first be analysed, extracted, and then inspected to ascertain how the parts fit together. This might involve obtaining the source code and relevant design documentation for analysis in software reverse engineering. Additionally, using tools like a disassembler to separate the software into its component parts may be necessary.

- Disassemblers (e.g. IDA Pro): An application will be disassembled to create assembly code. Although they are not accessible for all architectures, decompilers are also available for turning binary code into native code.
- Debuggers (e.g. x64dbg, Windbg, GDB): Reversers modify a program's execution with the aid of debuggers to learn more about what it is doing while it is executing. Additionally, they enable the engineer to manage portions of the program's memory while it is in operation. This makes it possible to see more clearly what the software is doing and how it affects a system or network.
- Modeling: The gathered data is abstracted into a conceptual model, each component of which explains its role in the wider framework. Taking information unique to the original and abstracting it into a general model that can be used to direct the creation of other things or systems is the goal of this stage. An example of this in software reverse engineering could be a data flow diagram or a structure chart.
- Analyzers for networks (e.g. Wireshark): Engineers can use network analyzers to learn how a software interacts with other computers, including the connections it is trying to make and the data it is trying to deliver.
- Review: To check that the model is a realistic abstraction of the original object or system, review it and test it in various circumstances. Software testing may be used to accomplish this in software engineering. The model can be used to redesign the original thing after it has been tested.

6. CONCLUSION

Cybersecurity experts can mitigate advanced malware faster and more effectively by using dynamic analysis to automate as much of the malware analysis as they can. This frees up their time for the really challenging tasks, like learning new encryption schemes, reverse communication protocols, or working on attribution. The more sophisticated the automated solution, the more likely it is that a reverser won't need to return to the process's first (and time-consuming) step, which involves unpacking, deobfuscating, and comprehending the primary actions of the malware.

In reality, cybersecurity teams must use a two-pronged strategy where sandbox technologies are employed to automatically analyse the vast majority of threats and reversers devote their time to surgically analyse the internals of the most sophisticated ones when additional threat intelligence is required.

REFERENCES

[1] Fontana, I. (2022). The human (in) security trap: how European border (ing) practices condemn migrants to vulnerability. International Politics, 59(3), 465-484.

[2] Li, Y., & Liu, Q. (2021). A comprehensive review study of cyber-attacks and cyber security; Emerging trends and recent developments. Energy Reports, 7, 8176-8186.

[3] Roseline, S. A., & Geetha, S. (2021). A comprehensive survey of tools and techniques mitigating computer and mobile malware attacks. Computers & Electrical Engineering, 92, 107143.

[4] Lukings, M., & Habibi Lashkari, A. (2022). Cybersecurity and Cybercrimes. In Understanding Cybersecurity Law and Digital Privacy (pp. 59-96). Springer, Cham.

[5] Heartfield, R., & Loukas, G. (2018). Detecting semantic social engineering attacks with the weakest link: Implementation and empirical evaluation of a human-as-a-security-sensor framework. Computers & Security, 76, 101-127.

[6] Afreen, A., Aslam, M., & Ahmed, S. (2020, October). Analysis of fileless malware and its evasive behavior. In 2020 International Conference on Cyber Warfare and Security (ICCWS) (pp. 1-8). IEEE.

[7] Or-Meir, O., Nissim, N., Elovici, Y., & Rokach, L. (2019). Dynamic malware analysis in the modern era—A state of the art survey. ACM Computing Surveys (CSUR), 52(5), 1-48.

[8] Tabrizchi, H., & Kuchaki Rafsanjani, M. (2020). A survey on security challenges in cloud computing: issues, threats, and solutions. The journal of supercomputing, 76(12), 9493-9532.

[9] Yan, P., & Yan, Z. (2018). A survey on dynamic mobile malware detection. Software Quality Journal, 26(3), 891-919.

[10] Alshamrani, A., Myneni, S., Chowdhary, A., & Huang, D. (2019). A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. IEEE Communications Surveys & Tutorials, 21(2), 1851-1877.

[11] Niknejad, N., Ismail, W., Ghani, I., Nazari, B., & Bahari, M. (2020). Understanding Service-Oriented Architecture (SOA): A systematic literature review and directions for further investigation. Information Systems, 91, 101491.